Digital Sensitivity: Predicting Signal Interaction using Functional Analysis*

Desmond A. Kirkpatrick Alberto L. Sangiovanni-Vincentelli

Department of EECS, University of California, Berkeley, CA 94720

Abstract

Maintaining signal integrity in digital systems is becoming increasingly difficult due to the rising number of analog effects seen in deep sub-micron design. One such effect, the signal crosstalk problem, is now a serious design concern. Signals which couple electrically may not affect system behavior because of timing or function in the digital domain. If we can isolate observable coupling effects then we can constrain layout synthesis to eliminate them [1]. In this paper, we find that it is possible to predict signal interaction by signal functionality alone, leading to a significant amount of robust switching isolation, independent of parasitics introduced by layout or semiconductor process. We introduce techniques to predict signal interaction using functional sensitivity analysis. In general sequential networks we find that significant switching isolation can be extracted with efficient sensitivity analysis algorithms, thus giving promise to the goal of synthesizing layout free from crosstalk effects.

1 The Crosstalk Problem

With the advent of sub-micron design, today's designers deal increasingly with analog effects arising from aggressive semiconductor processes and circuits. These effects violate signal integrity, something not normally a central part of digital design and hence few tools exist in the digital domain to help manage this problem.

Aggressive semiconductor process scaling is leading to an increase in wire coupling. Unlike transistor devices, the performance of interconnect does not scale well into deep sub-micron process dimensions. Several measures of interconnect performance scale inversely with wire cross-section, including increased resistance (delay) and current density (electro-migration). Wire cross-section is being improved through non-uniform scaling (using wires with higher aspect ratio), but this increases wire coupling.

Aggressive circuit design techniques, increasingly prevalent in high performance designs, include pre-charge circuitry, domino logic, and low voltage swing signals, all of which are sensitive to coupling noise. Many of these circuits trade "noise margin" for increased speed. Thus, these techniques increase susceptibility to crosstalk failures.

Until now, crosstalk noise has been managed within the noise margin of level-restoring logic, maintaining the digital design abstraction. A fundamental design assumption is that digital signals do not interact; there has been little development of methodology or tools to address coupling.

In the analog domain, low noise is the primary figure of merit, and shield wires are used to maintain signal integrity of sensitive nodes. Shields work against the goal of digital process scaling: to place circuits closer together, minimizing wire-length to gain performance. While there exist analog synthesis tools which handle crosstalk, the complexity of digital systems requires a different approach.

*This project is supported by the Semiconductor Research Corporation under grant 96-DC-324. Its support is gratefully acknowledged.

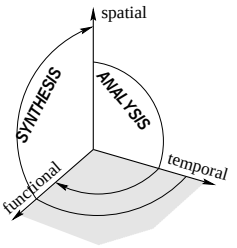


Figure 1: Signal Interaction Space

1.1 Digital Noise

Coupling noise in the digital domain has characteristics of which we can take particular advantage. Crosstalk is due to signal switching, thus every signal can be an *aggressor* as well as a *victim* of coupling noise. Crosstalk also causes digital systems to fail rather than degrade. A charge-storage node can be accidentally discharged through capacitive coupling, which we classify as a *logic effect*. Coupling can induce additional delay, causing a logic path to exceed the design clock period, which we call a *delay effect*. In both cases, a signal can tolerate a certain amount of noise before onset of the effect; these are threshold effects. During certain time periods or logic conditions digital circuits can withstand large amounts of noise. Finally, we note that digital circuit noise is not random, as it is caused by switching events. A switching event induces directional noise (e.g. increasing or decreasing voltage) which has associated timing and logic conditions. We can take advantage of crosstalk noise thresholds, direction, timing, and logic conditions in identifying observable noise effects.

2 Approach

Figure 1 illustrates three signal interaction axes: *temporal*, *functional*, and *spatial* (physical adjacency). We envision two major paradigms for managing signal integrity: post-layout analysis and constraint-driven synthesis, illustrated by the arcs in the figure.

From layout (spatial information), one can identify potential wire coupling problems due to long wire adjacencies and then filter these by considering temporal and functional interaction to see if they induce a logical failure. Such an analysis path is one taken by [3] using a crosstalk 'fault' analysis tool. Failures can then be fixed by re-routing the interacting signals.

The other, more powerful paradigm is to first analyze signals for interaction and then use this analysis to guide layout synthesis tools such as our channel router [1] to produce layout free from crosstalk effects. A system of con-

straints is necessary because of the threshold nature of coupling effects in digital systems (as opposed to introducing an objective function, such as that used in optimization for power or area). Constraint-driven synthesis leads to a correct-by-construction result, requiring no layout iterations to fix coupling problems. However, the number of coupling terms in a digital system is potentially immense and we propose taking advantage of the nature of digital noise to cope with this complexity.

2.1 Digital Sensitivity

Considering interaction between signal pairs, we say victim v is *digitally sensitive* to aggressor a if a can observably affect v's system behavior. We represent this as D, a 0-1 sensitivity matrix. Digital sensitivity annihilates many coupling terms and dramatically simplifies constraints.

2.1.1 Constraint System

Our constraint system is similar to that from an analog synthesis paradigm developed in [7]. For coupling parasitics, let N_v represent the coupling noise induced on victim node v. Let q_{va} represent the parasitic physical coupling between v and aggressor a, introduced by the layout process. We assume each aggressor swings the same amount of voltage and so the amount of noise injected onto v is determined only by the degree of physical coupling. Here, we define node v's noise sensitivity to this parasitic.

$$\begin{aligned} S_{va} &= \frac{\delta N_v}{\delta q_{va}} = A_{va} D_{va} \\ N_v &= \sum_a S_{va} q_{va} \leq B_v \end{aligned}$$

Here B_v is a node v's allowed threshold of noise and A represents the analog or circuit level interaction between node pairs, given by circuit analysis.

2.1.2 Layout Synthesis

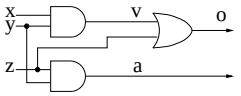
Our previous work in constraint-driven crosstalk routing [1] shows the trend that fewer shields are needed with increasing digital isolation, and that about 50% isolation is needed to have reasonable impact on final circuit area.

3 Digital Sensitivity Analysis

Digital sensitivity analysis considers interaction between signal pairs at a logical, rather than electrical level. The goal of sensitivity analysis is to extract as much signal isolation information as possible, employing approximations for efficiency. The critical question is what amount of functional isolation can be efficiently extracted from networks.

3.0.3 Functional Digital Sensitivity

In this paper, we focus on predicting signal interaction based on functionality in both combinational and sequential networks. In each domain, we first introduce analysis of the functional transitions, i.e., the switching behavior as predicted by the function at each node of the network. Functional transition analysis is useful both as an intuitive illustration and more importantly, for specific types of circuitry. For general CMOS circuitry, we must also consider signal glitching behavior, which we introduce in Section 4.4.4. We utilize a *signal transition network* to model complete signal behavior. We present four domains and methods of analysis: combinational logic, functional transition (CF); combinational logic, glitching transition (CG); sequential logic, functional transition (SF); and sequential logic, glitching transition (SG). Thus we build a complete approach to functionally predicting signal interaction.



$$\begin{array}{rclcrcl} v & = & xy & & a & = & yz \\ v^1 & = & \overline{z}xy & & \overline{v^0} & = & z + xy \end{array}$$

Figure 2: Unobservable Signal Interaction

4 Combinational Analysis

In this section we introduce functional sensitivity analysis for signal interaction in combinational networks, first for the pure functional transition, and then including interaction due to signal glitches in logic with hazards.

Consider the circuit in Figure 2. We can find a pair of input vectors for primary inputs $\{x, y, z\}$ which cause signal a to rise while v falls. v's descent may be slowed if v and a are physically adjacent in layout. However, in order to *observe* this slowing at primary output o, z must be low. If z is low, a cannot rise, and so this particular interaction is unobservable. We formalize this analysis, including don't care conditions, in the following sections.

4.1 Basic Definitions

Definition 1 A *completely specified Boolean function* f with n inputs and l outputs is a mapping $f : B^n \rightarrow B^l$ where $B = \{0, 1\}$.

Definition 2 Let $A \subseteq B^n$. The *characteristic function* of A is the function $f : B^n \rightarrow B$ defined by $f(m) = 1$ if $x \in A$, $f(m) = 0$ otherwise.

Definition 3 An *incompletely specified Boolean function* F with n inputs and l outputs is a mapping $F : B^n \rightarrow Y^l$, $Y = \{0, 1, *\}$. F's onset, offset, and don't care set are:

$$\begin{aligned} f^1 &= \{m \in B^n \mid f(m) = 1\} \\ f^0 &= \{m \in B^n \mid f(m) = 0\} \\ f^{DC} &= \{m \in B^n \mid f(m) = *\} \end{aligned}$$

Definition 4 Let $f : B^n \rightarrow B$ be a Boolean function and x_i an input variable to f. The *cofactor* of f with respect to a literal $x_i(\overline{x_i})$ denoted $f_{x_i}(f_{\overline{x_i}})$ is a new function obtained by substituting 1(0) for $x_i(\overline{x_i})$ in every cube in f which contains $x_i(\overline{x_i})$.

Definition 5 The *observability don't cares (ODCs)* at each node y of a multi-level logic network N are conditions under which y can be either 0 or 1 while the functions generated at the primary outputs remain unchanged. Denoting primary outputs $z = \{z_1, ..., z_l\}$, the complete ODC at node y is:

$$ODC(y) = \{m \in B^n | z_y(m) \equiv z_{\overline{y}}(m)\}$$

Definition 6 A *cover* for the incompletely specified Boolean function F: $B^n \rightarrow Y$ is any completely specified function f such that $f(m) = 1$ if $F(m) = 1$, $f(m) = 0$ if $F(m) = 0$, and $f(m) = 0$ or 1 if $F(m) = *$.

4.2 Coupling Don't Care Conditions

The ODCs of an intermediate node in a logic network give flexibility to how that node is implemented (i.e. what cover the logic induces). We can view ODCs as allowing some perturbation of the logic function of an intermediate node v (in our case, due to coupling noise). If this perturbation remains within $ODC(v)$, then it is unobservable, and so we can consider v to be functionally insensitive to this perturbation.

We may have multiple simultaneous noise sources to consider. In the analog domain, we use a linearity approximation for multiple sources, which enables superposition. In the digital domain, we parallel this approximation by requiring that don't care sets are compatible, so that digital sensitivity obeys the superposition principle. Thus, we define offset v^0 and onset v^1 using Compatible Observability Don't Cares as described in [8]. Note that since CODCs are not canonical, this definition of digital sensitivity is therefore not canonical.

$$\begin{aligned} v^0 &= \overline{v} \cdot \overline{CODC(v)} \\ v^1 &= v \cdot \overline{CODC(v)} \end{aligned}$$

4.2.1 Signal Transition Sets

When a victim node v switches under the same conditions as a potential aggressor, we are concerned if we can observe the *final* circuit state of the victim node as distinct from it's intended *final* logic state. If these states are indistinguishable because of logic network don't cares, we consider this a don't care signal transition.

Definition 7 Let $p, q \in \{0, 1\}$ be signal states. A *signal transition set* a^{pq} is a mapping $a^{pq} : I \times I \rightarrow B$ representing the characteristic function of the input vector pairs in $I \times I$ which cause signal a to transition from signal state p to q. Note: $\overline{a^{10}} = a^{00} + a^{01} + a^{11}$.

Definition 8 A *care signal transition set*, denoted $c(v^{pq})$, is a mapping $c^{pq} : I \times I \rightarrow B$ representing the subset of signal transitions v^{pq} satisfying $CODC(v)$ in the final state q as follows:

$$c(v^{pq}) = (\{1\} \times \{v^q\}) \cap v^{pq}$$

4.3 Functional Transition Analysis (CF)

Suppose our victim transition v^{lm} is sensitive to adjacent signal transitions from p to q. Suppose further that under the transitions of interest our network is hazard-free. Then we can define digital sensitivity with respect to functional transitions v^{lm} and a^{pq}:

Definition 9 Let $D_{va}^{pq,lm}$ be the *digital sensitivity (functional transition)* of victim v with respect to aggressor a when v is transitioning from state p to state q while a interferes by transitioning from state l to m.

$$D_{va}^{pq,lm} = \begin{cases} 0 & \text{if } c(v^{pq}) \cdot a^{lm} = \emptyset \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

General signal interactions can be captured using this functional sensitivity formulation. Consider the special delay effect where a victim settles to signal state 0 only to have an upward spike injected as it is being sampled by a latch. This can be represented by $D_{va}^{X0,01}$.

Definition 10 For delay effects, where $p = m$, $q = l$, and $p \neq q$, let D_{va}^{pq} denote $D_{va}^{pq,qp}$. Furthermore, let $D_{va}^{\uparrow} = D_{va}^{01}$ and $D_{va}^{\downarrow} = D_{va}^{10}$.

In hazard-free networks, delay effect isolation can be formulated using pre- and post-conditions:

$$D_{va}^{\uparrow} = \begin{cases} 0 & \text{if } a \subseteq v + v^1 \subseteq a \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

$$D_{va}^{\downarrow} = \begin{cases} 0 & \text{if } v \subseteq a + a \subseteq \overline{v^0} \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

Note that if $v \subseteq a$ and $a = 1$, then $v = 1$, so a cannot oppose v falling; $v \subseteq a$ is a general isolation precondition for $D_{va}^{\downarrow}$. If we recall the circuit in Figure 2, we were concerned with this case of a slowing v's downward transition. In fact, if we consider the postcondition for $D_{va}^{\downarrow} = 0$, don't cares on v prevent a's effect on v's downward transition from be observed:

$$\begin{aligned} a \subseteq \overline{v^0} &\Rightarrow D_{va}^{\downarrow} = 0 \\ a \not\subseteq v, \quad v^1 \not\subseteq a &\Rightarrow D_{va}^{\uparrow} = 1 \end{aligned}$$

4.3.1 Method CF

Method CF is the use of Equations 2 and 3 on combinational networks for analyzing functional transitions only. Many noise-sensitive circuits, such as asynchronous or self-timed logic, as well as pre-charge or domino circuitry, also have the property that their logic is hazard-free. Thus the analysis we have presented for functional transition interaction is directly applicable to these circuits.

Functional transition analysis is quite simple in the combinational case: here we simply use set containment tests which involve functions of the network such as a, v, v^0, v^1 and their complements. We use Binary Decision Diagrams to represent all functions and set computations. We apply the formula in Definition 9 to compute functional transition interaction.

4.4 Full Sensitivity Analysis (CG)

General CMOS combinational logic, on the other hand, has hazards which increase signal interaction problems as the glitches they produce cause switching noise in both up and down directions. Furthermore, glitches are difficult to predict accurately and are often over-estimated, potentially restricting functional sensitivity techniques to hazard-free networks. In this section, we strive to capture complete switching behavior of CMOS logic implementations in digital sensitivity analysis.

4.4.1 Hazard Definitions

Definition 11 Let transition cube $[A, B]$ be all minterms that can be reached when transitioning from A to B. A Boolean function contains a *static hazard* for inputs from A to C iff $f(A) = f(C)$, and there exists some state $B \in [A, C]$ such that $f(A) \neq f(B)$.

Definition 12 A Boolean function contains a *dynamic hazard* for inputs from A to $D \iff f(A) \neq f(D)$ and there exists a pair of states B and C ($A \neq B$, $C \neq D$ such that $B \in [A, D]$, $C \in [B, D]$, $f(B) = f(D)$, and $f(A) = f(C)$.

Definition 13 A combinational circuit for a function f contains a *static logic hazard* for the input transition from A to $B \iff f(A) = f(B)$ and for some delay assignment, the circuit's output changes momentarily during the transition interval.

Definition 14 A combinational circuit for a function f contains a *dynamic logic hazard* for the input transition from A to $B \iff f(A) \neq f(B)$ and for some delay assignment, the circuit's output is not monotonic during the transition interval.

Definition 15 A *controlling value* at a gate input is the value that determines the value at the output of the gate input independent of other inputs. A *non-controlling value* at a gate input is a value which is not controlling.

4.4.2 Digital Sensitivity with Hazards

Definition 16 A *glitchy signal transition set*, denoted $g(a^{pq})$, is a mapping $g^{pq} : I \times I \rightarrow B$ representing the subset of signal transitions a^{pq} which are hazardous (either static or dynamic). A *stable signal transition set*, denoted $s(a^{pq})$, is a mapping $s^{pq} : I \times I \rightarrow B$ representing the subset of signal transitions a^{pq} which are hazard-free. Glitchy and stable signal transition sets have the following relationship:

$$\begin{aligned} s(a^{pq}) &= a^{pq} \cdot \overline{g(a^{pq})} \\ g(a^{pq}) &= a^{pq} \cdot \overline{s(a^{pq})} \end{aligned}$$

Definition 17 Let $D_{va}^{pq,lm}$ be the *digital sensitivity* of victim v with respect to aggressor a when v is transitioning from state p to state q while a interferes by transitioning from state l to m.

$$D_{va}^{pq,lm} = \begin{cases} 0 & \text{if } c(v^{pq}) \cdot (a^{lm} + g(a^{**})) = \emptyset \\ 1 & \text{otherwise} \end{cases}$$

4.4.3 0-delay Stability

As we are analyzing logic networks in preparation for layout synthesis, circuit delays are not accurately known. However, predicting circuit glitching typically requires knowledge of gate delays. Here, we present a 0-delay formulation of glitch prediction which is very pessimistic: it may predict glitches that can only occur with non-causal circuit behavior. While conservative, this formulation predicts signal stability which is valid for any gate and wire delay assignment. We compute the noise due to glitching $g(a^{pq})$ using a recursive formulation of $s(a^{pq})$. Inverters introduce no hazards, so their stability formula is simple:

$$INV: \qquad s(a^{pq}) = s(y^{\overline{p}\ \overline{q}}) \tag{4}$$

For AND or OR gates, we formulate signal stability recursively. Let a represent the output of the gate and x_i represent the i-th input. For the inputs, denote controlling and non-controlling values c and n respectively: AND: $c = 0, n = 1$, OR: $c = 1, n = 0$. $*$ represents an input don't care. Signal stability is defined as follows:

$$s(a^{cc}) = \sum_i s(x_i^{cc}) \tag{5}$$

$$s(a^{nn}) = \prod_i s(x_i^{nn}) \tag{6}$$

$$s(a^{nc}) = \sum_i s(x_i^{nc}) \prod_{j \neq i} s(x_j^{n*}) \tag{7}$$

$$s(a^{cn}) = \sum_i s(x_i^{cn}) \prod_{j \neq i} s(x_j^{*n}) \tag{8}$$

Here we have extended the stable low- and stable high-value Equations 5 and 6 as defined in [5] to include stable *transitions* (Equations 7 and 8). We assume inputs are stable: $(s(i^{**}) = 1$ for all input variables i.

4.4.4 Method CG

Method CG is the application of the full digital sensitivity (Definition 17) to combinational networks that have hazards. We incorporate 0-delay signal stability (Section 4.4.3) to include levelized timing, omitted for brevity.

For full signal behavior analysis, we implement the computation of transition sets a^{lm} and v^{pq} using a *combinational transition network* composed of two copies of the original combinational logic network. Each network $CL(P)$ and $CL(N)$ computes the response in the first and second input vector spaces P and N respectively. To compute $c(v^{pq})$ using Equation 8, we use the CODC extraction techniques of [8]. We compute the stability sets $s(a^{pq})$ recursively using the formulas in Section 4.4.3. We apply the digital sensitivity formula in Definition 17 to compute combinational functional sensitivity.

5 Sequential Analysis

Analyzing combinational logic alone will predict signal interactions which cannot occur in real machines. This is because combinational logic is used to implement state machines in digital systems. State machines, by their very nature, restrict the input transitions that the combinational logic sees. Additionally, they have an associated initial state and a transition graph, limiting the input transitions to the logic even further, to the *reachable* set of transitions.

5.1 Set Projections and Images

Definition 18 Let $f : B^n \rightarrow B$ be a Boolean function and $x = \{x_1, ..., x_k\}$ be a subset of the input variables. The *existential quantification (smoothing)* of f by x is:

$$\begin{aligned} \exists_{x_i} f &= f_{x_i} + f_{\overline{x_i}} \\ \exists_x f &= \exists_{x_i} ... \exists_{x_k} f \end{aligned}$$

Definition 19 Let $f : B^n \rightarrow B$ be a Boolean function with *support* $y = \{y_1, ..., y_k\}$. Let $x = \{x_1, ..., x_k\}$ be another subset of variables, describing another subspace of B^n of the same dimension k. The *substitution* of variables y by variables x in f is the function of x obtained by substituting x_i for y_i in f:

$$f_{|_{y=x}} = f(x) \qquad \text{if } x_i = y_i \text{ for all } 1 \leq i \leq k$$

Definition 20 Let $f : B^n \rightarrow B^m$ be a Boolean function and A a subset of B^n. The inverse image of A by f is the set $f^{-1} = \{x \in B^n | f(x) = y,\ y \in A\}$.

5.2 Functional Transition Analysis (SF)

Definition 21 A Finite State Machine is a quintuple (I, O, Q, T, q_0) where I is the (true) input space, O is the output space, Q is set of states, with $q_0 \in Q$ as the initial state, and T is the transition relation, a subset of $I \times Q \times Q$. We consider only deterministic finite state machines: $n = T(i, p), n, p \in Q, i \in I$. Let $R \subseteq Q$ denote the set of *reachable states* of the machine.

We note that digital sensitivity is equivalent to whether two transition sets such as a^{pq} and $c(v^{qp})$ intersect. We can reformulate using pre-conditions $((a = p) \cdot (v = q))$ and post-conditions $((a = q) \cdot v^p)$ as follows:

$$\begin{aligned} pre(i, p) &= pre_v(i, p) \cdot pre_a(i, p) \cdot R(p) \\ post(ni, n) &= post_v(n, n) \cdot post_a(ni, n) \end{aligned}$$

$R(p)$ restricts analysis to *reachable* transitions. Variables ni in the post condition to reflect the fact that inputs may change with the state transition.

Definition 22 Let D_{va} be the *digital sensitivity* in the sequential domain:

$$D_{va} = \begin{cases} 0 & \text{if } pre(i, p) \cdot T(i, p, n) \cdot \exists_{ni} post(ni, n) = \emptyset \\ 1 & \text{otherwise} \end{cases}$$

To compute D, we image either the pre- or post-condition using the next state function. We choose to formulate this in terms of a reverse or *pre-image* step (using a one-step BDD variable substitution):

$$D_{va} = \begin{cases} 0 & \text{if } \{\exists_i post(i, p)\}_{|_{p=n}} \cap pre(i, p) = \emptyset \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

5.2.1 Pre-Image Approximation

The pre-image step is efficient because it smoothes away inputs to the post-condition before substituting the state variables for next state functions; primary inputs are the support for all BDDs. However, this pre-image computation must occur for all node pairs, even if they have disjoint 'true' inputs, which are smoothed away before the pre-image. A *pre-image approximation* comes from realizing that

$$\{\exists_i post(i,p)\}_{|_{p=n}} \subseteq \{\exists_i post_a(i,p)\}_{|_{p=n}} \cap \{\exists_i post_v(i,p)\}_{|_{p=n}}$$

and that equality holds if v and a have disjoint support in true inputs i. Thus, for every node a we can pre-compute the pre-image in the domain of original primary inputs. We use the pre-image approximation as follows:

1. if pre-image approximations are disjoint, $D_{va} = 0$
2. else if a and v have disjoint true inputs then $D_{va} = 1$
3. else perform the exact pre-image

5.2.2 Method SF

Method SF is the application of Equation 9 to compute sequential sensitivity for functional transition interaction. This is applicable to sequential logic networks which are hazard-free under the reachable transitions of the system, such as asynchronous FSMs or synchronous systems constructed from domino logic.

Analyzing the interaction of functional transitions in sequential circuits can be accomplished using sets in $I \times Q$. The pre-image computation, with appropriate smoothing, can be accomplished in the same domain. This fact keeps the BDDs small and the computation fast by minimizing the number of support variables. Thus Method SF is quite efficient.

5.3 Full Sensitivity Analysis (SG)

The glitching behavior of combinational logic tends to increase signal interaction, while the nature of sequential logic tends to restrict signal interaction and observability.

Figure 4 is the tiny shift register example from the MCNC 1991 FSM benchmarks. Suppose we are concerned with a delay effect on the downward transition of victim v. Sequential digital sensitivity analysis with reachability predicts that potential aggressor a never transitions upward when v transitions downward. However, a can glitch when $s1 = 1$, injecting noise onto v, as shown by the timing diagram. Yet, if we include ODC analysis, we see that $s1 = 1$ implies that any noise on v cannot be observed, and so this is an unobservable signal interaction. Our analysis tool predicts, in fact, that $D^{\downarrow}_{va} = 0$ for this circuit.

5.3.1 Method SG

Method SG is the application of Definition 17, defined for combinational networks, to sequential hazardous circuits.

For full signal analysis of sequential logic, we use a *signal transition network* composed of two copies of the combinational logic network of the original machine. As in the combinational case, each network $CL(P)$ and $CL(N)$ computes the response in the first and second input vector spaces P and N respectively. However, for the sequential case, we connect the next-state outputs n of the first machine to the state inputs of the second machine, as illustrated in Figure 3. Effectively we have partially 'unrolled' the state machine one level. The support of the BDDs is kept small: the domain of all sets is $I \times Q \times NI$, where NI is the true primary input space of the sequential machine. The imaging step

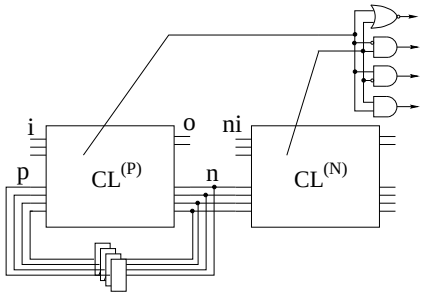


Figure 3: Sequential Transition Network

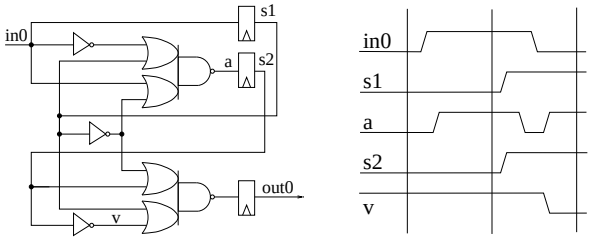


Figure 4: Sequential Example: Shiftreg

is implicit in the network construction. The pre-image approximation has a similar approximate network construction.

We compute the reachable states R for both Method SF and SG using the techniques of [2].

6 Results

In Table 1 we show the isolation measurements of a set of ISCAS89 sequential benchmarks, synthesized using SIS's 'script.rugged' script and mapped for minimum delay. For each example, we list its name, number of inputs i, latches l, and nodes n. The measurements are the percentage of signal pairs which are isolated (insensitive) from UP noise, considering each signal as a victim. CPU time in seconds is for a 300MHz DEC Alpha processor. For glitching measurements, we assume a unit-delay model, assigning min/max levels to nodes and analyze using a timed version of the signal stability equations in Section 4.4.3.

For combinational logic, we note that Method CF analyzes the functional transitions including ODCS (first column) quite fast, with a resulting average isolation of about 15%. Method CG in column two includes glitching behavior and here we see the isolation numbers drop dramatically. Method SF in the third column shows that functional transitions in sequential systems are well isolated (about 40%). Note that including reachable states does not significantly increase isolation. Method SG, in the fourth column illustrates a most interesting result that adding glitching behavior does not reduce isolation (here, 28%) nearly as much as in the combinational case. We assign this to the combination of the transition function restricting input transitions (controllability) and the ODCs restricting observability of signal interactions.

Finally, we note that the computation time of Method SF is aided a great deal by the pre-image approximation. In Table 2 the first column shows that the approximation algorithm yields isolation numbers quite close the exact results in the second column, but with dramatically smaller CPU times. The final column shows the effect of using the approximation algorithm as a filter.

Table 1: Functional Analysis Results

Example				Combinational				Sequential					
				Method CF		Method CG		Method SF		Method SF		Method SG	
				ODCs		Glitch ODCs		DCs		Reach ODCs		Reach / Glitch ODCs	
	i	l	n	ISO(%)	CPU	ISO(%)	CPU	ISO(%)	CPU	ISO(%)	CPU	ISO(%)	CPU
s344	15	9	123	12.2	0.7	6.0	1.5	31.2	2.0	37.1	5.0	17.8	8.1
s349	15	9	123	12.9	0.6	6.0	1.3	30.7	2.5	37.5	5.0	18.4	8.5
s382	21	3	151	9.8	1.0	2.0	1.8	35.3	3.2	41.1	9.4	30.1	5.7
s386	6	7	103	27.2	0.6	7.1	1.6	52.0	1.6	58.7	1.7	36.5	1.4
s400	21	3	145	9.4	0.9	2.0	1.6	33.6	2.6	40.0	9.8	29.2	5.5
s420	16	18	146	23.6	1.6	-	space	60.1	3.1	60.1	94.1	40.7	89.7
s444	21	3	139	8.2	0.9	1.8	1.7	33.1	2.7	38.7	9.3	27.9	5.8
s510	6	19	239	28.1	2.4	7.4	5.2	62.0	3.1	67.1	3.3	47.5	3.9
s526	21	3	138	8.2	0.9	2.2	1.5	30.3	1.9	38.7	8.7	26.6	6.2
s641	17	35	160	5.7	3.2	1.1	42.4	13.8	53.8	35.6	112.7	30.3	144.8
s713	17	35	157	5.6	3.1	1.1	35.9	13.4	52.8	35.6	131.1	30.3	157.6
s820	5	18	277	30.3	4.3	7.1	9.9	51.5	11.9	53.7	13.7	32.8	9.1
s832	5	18	275	28.3	5.0	6.1	10.7	47.0	16.1	48.4	18.3	26.6	10.1

Table 2: Pre-Image Approximation Results

Example	Approx		PreImage		Filter
	ISO(%)	CPU	ISO(%)	CPU	CPU
s344	30.9	1.2	31.2	13.7	2.0
s349	30.5	1.4	30.7	17.2	2.4
s382	31.4	1.0	35.3	10.3	5.7
s386	44.5	0.5	52.0	1.7	1.1
s400	29.9	0.9	33.6	6.9	3.5
s420	58.6	0.8	60.1	4.6	2.3
s444	29.5	0.9	33.1	7.8	4.0
s510	61.9	1.6	62.0	10.0	2.5
s526	28.1	0.9	30.3	11.1	2.9
s641	10.1	1.9	13.5	21.0	17.8
s713	9.9	1.9	13.4	19.6	16.6
s820	48.8	2.4	51.5	13.7	7.9
s832	43.8	2.6	47.0	16.2	10.0

7 Conclusions

Our results show that the level of switching isolation we can efficiently extract from logic networks is quite promising for constraining layout synthesis to avoid crosstalk effects. We are in the process of applying these techniques to industrial circuits where we have already found significant temporal isolation, and hope the combination will give us enough isolation to route large circuits free from crosstalk effects.

The ability to synthesize high-speed circuits free from critical crosstalk noise has several implications on design. Crosstalk delay effects are hard to predict or measure, making delay correlation between design and silicon quite difficult. Without digital sensitivity, designers could be overly cautious when shielding sensitive nodes of hand-designed high performance circuits. Moreover, signal integrity management could spread the use of these circuits to synthesis, for example in the control sections of microprocessors. The ability to manage crosstalk also has implications on the design of VLSI manufacturing processes. Global process optimization determines the feature sizes of interconnect layers from design performance tradeoffs, including wire aspect ratio. If we can manage crosstalk noise as part of design, process engineers can be more aggressive in increasing wire aspect ratio to optimize overall interconnect performance.

Digital sensitivity can also be applied to modern analog designs, such as switched capacitor filters, which have discontinuous waveforms not unlike digital systems, to manage signal integrity at a functional level.

Digital sensitivity is a new critical link between logic and physical design which will hopefully provide key insights into how these different design abstractions can be merged as digital circuits become more analog in nature.

References

[1] Desmond A. Kirkpatrick and Alberto L. Sangiovanni-Vincentelli. Techniques for crosstalk avoidance in the physical design of high performance digital systems. In *Proc. Int'l Conf. on Computer-Aided Design*, pages 616–619, November 1994.

[2] Hervé J. Touati, Hamid Savoj, Bill Lin, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli. Implicit state enumeration of finite state machines using BDDs. In *Proc. Int'l Conf. on Computer-Aided Design*, pages 130–133, November 1990.

[3] A. Rubio, N. Itazaki, X. Xu, and K.Kinoshita. An approach to the analysis and detection of crosstalk faults in digital vlsi circuits. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, 3(13):387–395, March 1994.

[4] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley, 1990.

[5] Debashis Bhattacharya, Prathima Agrawal, and Vishwani D. Agrawal. Test generation for path delay faults using binary decision diagrams. *IEEE Transactions on Computers*, 44(3):434–447, March 1995.

[6] M. Abramovici, M. Breuer, , and A. Friedman. *Digital systems testing and testable design.* IEEE Press, 1990.

[7] Henry Chang, Alberto L. Sangiovanni-Vincentelli, Felice Balarin, Edoardo Charbon, Umakanta Choudhury, Ganni Jusuf, Ed Liu, Enrico Malavasi, Robert Neff, and Paul R. Gray. A top-down, constraint-driven design methodology for analog integrated circuits. In *Proc. Custom Integrated Circuits Conf.*, pages 8.4/1–6, May 1992.

[8] Hamid Savoj. Don't cares in multi-level network optimization. *UCB Ph.D. Dissertation*, 1991.